Strategies for Acceleration of Deep Learning in Algorithm & CMOS-based Hardware

- V. Sze et. al., “Efficient Processing of Deep Neural Networks: A Tutorial and Survey”. Proc. IEEE Vol. 105, No. 12, Dec. 2017
- W. Dally, “High-Performance Hardware for Machine Learning”, NIPS 2015
- S. Han & W. Dally, “Deep Learning Tutorial and Recent Trends”, FPGA 2017
- A. Krizhevsky, “One weird trick for parallelizing convolutional neural networks”, arXiv, 2014
- J. Dean, “Machine Learning for Systems and Systems for Machine Learning”, NIPS 2017
- Etc..

Jan. 10. 2019
Jong Hoon Shin

Why Deep Learning Did Not Work in the Past?

Geoffrey Hinton summarized the findings up to today in these four points:

1. **Our labeled datasets were thousands of times too small.**

    Youtube comments , reCAPTCHA, Facebook privacy, Amazon reviews..

2. **Our computers were millions of times too slow.**

    GPU, FPGA, ASIC, Neuromorphic (Spiking, NVM)..

3. **We initialized the weights in a stupid way.**

    RBM, Xavier/Glorot/He initialization ..

4. **We used the wrong type of non-linearity.**

    ReLu series..

In A 'Brief' History of Neural Nets and Deep Learning, Part 4, http://www.andreykurenkov.com/writing/ai/a-brief-history-of-neural-nets-and-deep-learning-part-4/

Advances in Deep Learning Algorithms

Non-Linearities
Relu
Sigmoid
Tanh
GRU
LSTM
Linear
...

Optimizer
SGD
Momentum
RMSProp
Adagrad
Adam
Second Order (KFac)
...

Connectivity Pattern
Fully connected
Convolutional (Conv, MaxPooling)
Dilated
Recurrent
Recursive
Skip / Residual
Random

Loss
Cross Entropy
Adversarial
Variational
Max. Likelihood
Sparse
L2 Reg
REINFORCE
...

Hyper Parameters
Learning Rate
Decay
Layer Size
Batch Size
Dropout Rate
Weight init
Data augmentation
Gradient Clipping
Beta
Momentum

Scott Reed et. al., Deep Learning: Practice and Trends, NIPS2017

Advances in Tools for Deep Learning

Platforms

Microsoft Azure

aws

Google Cloud

Frameworks

scikit learn

Caffe

Keras

Microsoft CNTK

mxnet

theano

PYTORCH

Caffe2

Datasets

Caltech 101

IMAGENET

LDC Linguistic Data Consortium

UCI Machine Learning Repository

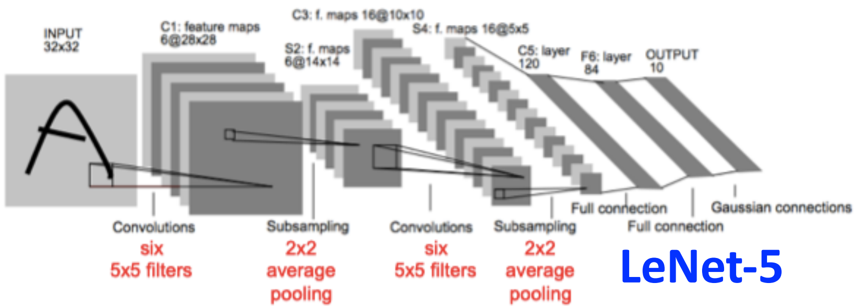
Center for Machine Learning and Intelligent Systems

WMT Workshop 2014

Scott Reed et. al., Deep Learning: Practice and Trends, NIPS2017

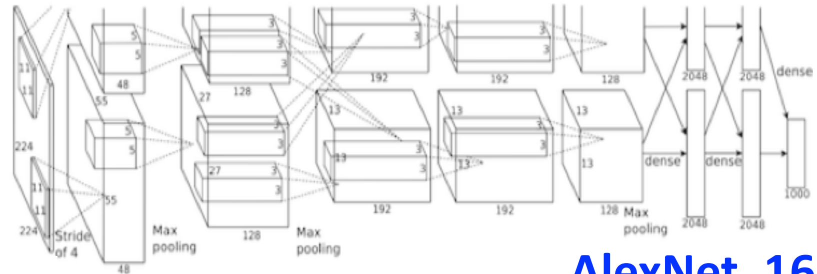
Famous DL Algorithms for MNSIT & ImageNet

MNIST

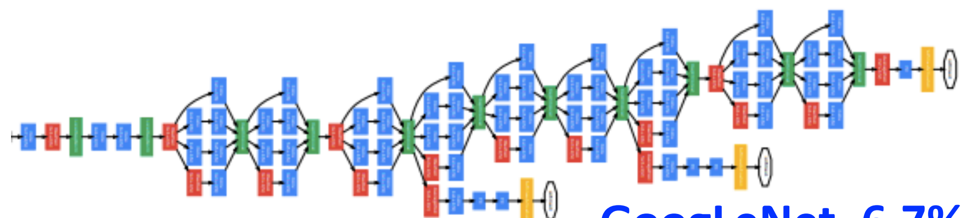


LeNet-5

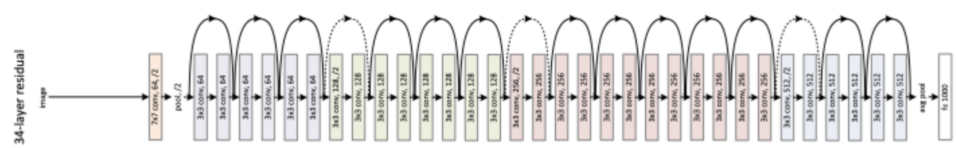
ImageNet Competition



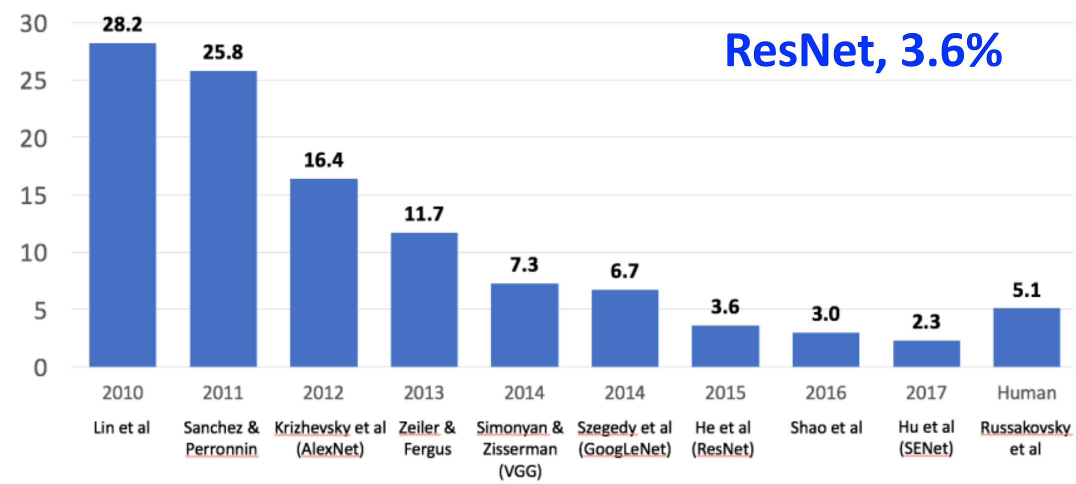
AlexNet, 16.4%



GoogLeNet, 6.7%

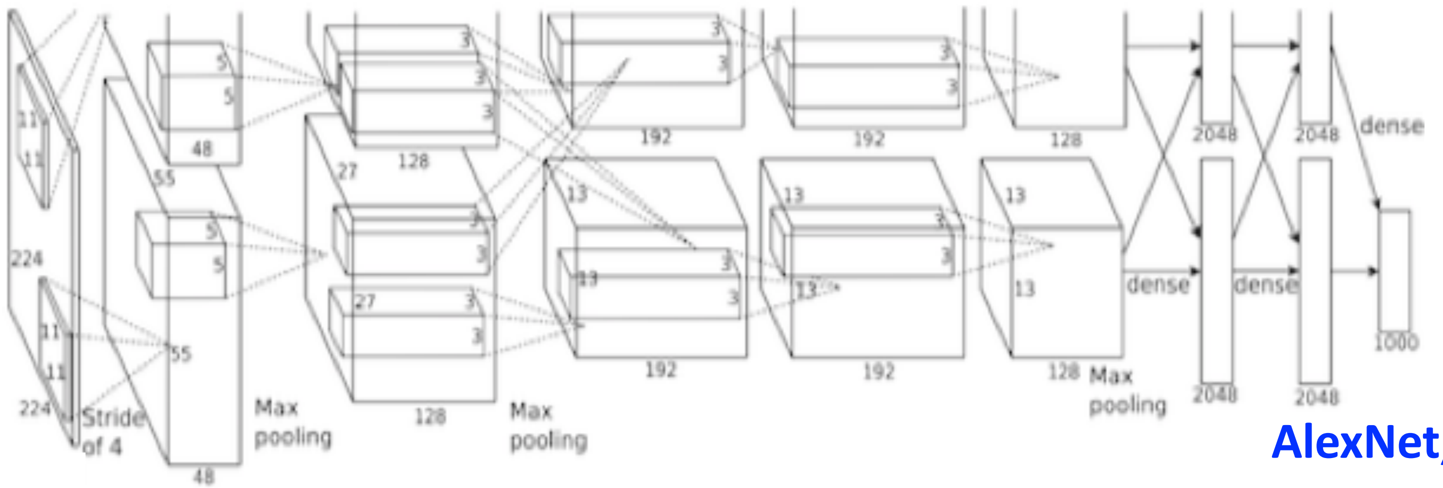


ResNet, 3.6%



- MNIST → 1998’
- ImageNet → 2018’
- Sequential data & Video(30fps)?
  - Throughput matters
  - # of width & layers
  - IOT/Edge → Energy matters

AlexNet's Parameters and MACs



AlexNet, 16.4%

Layer	Filter Size (RxS)	# Filters (M)	# Channels (C)	Stride
1	11x11	96	3	4
2	5x5	256	48	1
3	3x3	384	256	1
4	3x3	384	192	1
5	3x3	256	192	1

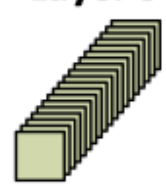
Layer 1

34k Params
105M MACs

Layer 2

307k Params
224M MACs

Layer 3



885k Params
150M MACs

Input Image: (224x224x3)

1st Output:
(224-11)/4+1 → 55X55

1st Params:
(11x11x3)x96 → 34K

1st MACs:
(34K x 55 x 55) → 105M

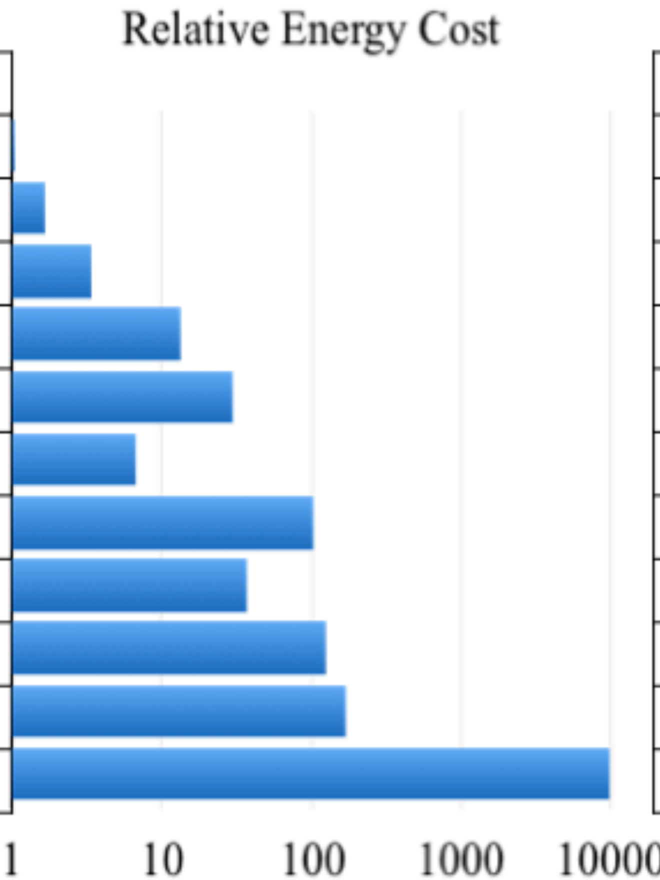
Summary of Parameters and MACs

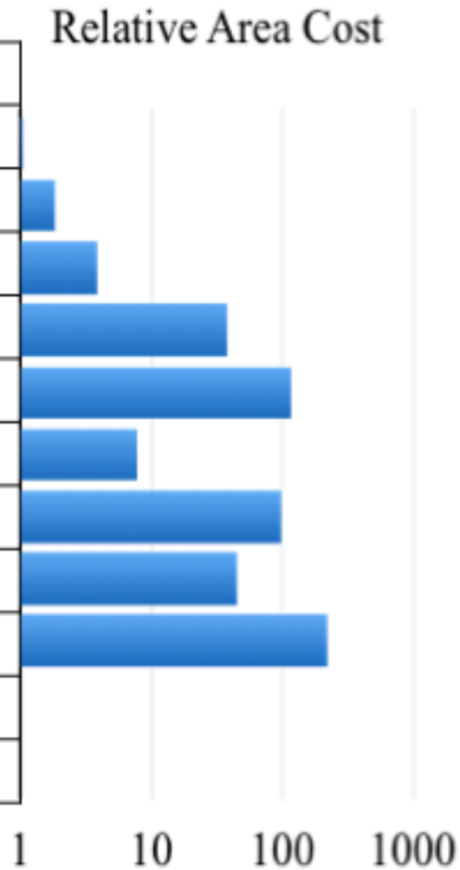
Metrics	LeNet-5	AlexNet	VGG-16	GoogLeNet (v1)	ResNet-50
Top-5 error	n/a	16.4	7.4	6.7	5.3
Input Size	28x28	227x227	224x224	224x224	224x224
# of CONV Layers	**2**	**5**	**16**	**21 (depth)**	**49**
Filter Sizes	5	3, 5,11	3	1, 3 , 5, 7	1, 3, 7
# of Channels	1, 6	3 - 256	3 - 512	3 - 1024	3 - 2048
# of Filters	6, 16	96 - 384	64 - 512	64 - 384	64 - 2048
Stride	1	1, 4	1	1, 2	1, 2
# of Weights	2.6k	2.3M	14.7M	6.0M	23.5M
# of MACs	283k	666M	15.3G	1.43G	3.86G
# of FC layers	**2**	**3**	**3**	**1**	**1**
# of Weights	58k	58.6M	124M	1M	2M
# of MACs	58k	58.6M	124M	1M	2M
Total Weights	**60k**	**61M**	**138M**	**7M**	**25.5M**
Total MACs	**341k**	**724M**	**15.5G**	**1.43G**	**3.9G**

- **Multiply 1.2M training image for a single training epoch!**
- **Most of Parameters → FC & Most of MACs→ Filter**

Considering Energy Consumption..

Operation:	Energy (pJ)	Area (μm^2)
8b Add	0.03	36
16b Add	0.05	67
32b Add	0.1	137
16b FP Add	0.4	1360
32b FP Add	0.9	4184
8b Mult	0.2	282
32b Mult	3.1	3495
16b FP Mult	1.1	1640
32b FP Mult	3.7	7700
32b SRAM Read (8KB)	5	N/A
32b DRAM Read	640	N/A





- Reduce **Memory Access**
- Reduce **Data Type : F→I, 32→16→8**
- Reduce **Mult & Add**

Mark Horowitz "Computing's Energy Problem (and what we can do about it)", ISSCC 2014
Prof. W. Dally, NIPS2015 tutorial

Strategies for DL Acceleration

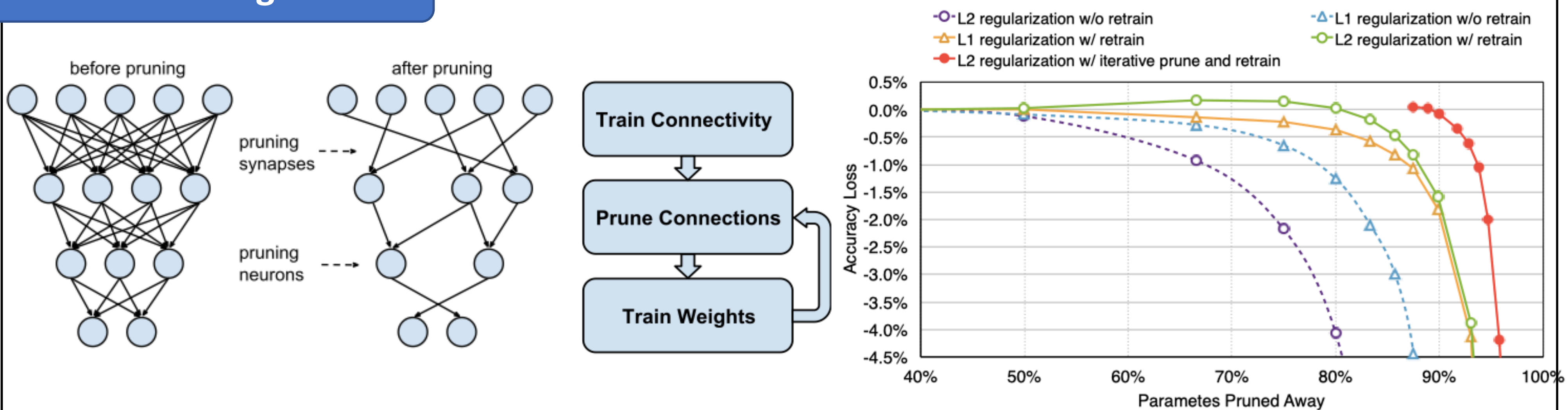
<table>
<tr><th></th><th>Inference</th><th>Training</th></tr>
<tr><td>Algorithm</td><td>Pruning : P-Threshold, +Training
Weight Sharing:
• w-KNN→Quant, Huffman coding
Quantization: Floating P.→ Fixed P.
Low Rank Approximation
Binary/Ternary Net
Computational Transform
• Winograd, FFT, Strassen</td><td>Parallelization:
• Data/Parameter/Hyp.Par
Mixed Precision MAC: FP16 + FP32
Model Distillation:
• Teachers(GGN,VGG,Res)&Student
DSD(Dense-Sparse-Dense) Training
Mixed Precision Net:
• BinaryConnect, QNN</td></tr>
<tr><td>Hardware</td><td colspan="2">Architecture:
• Temporal(SIMD,SIMT with Global Buffer) : General purpose GPUs
• Spatial(Local Memory in Dist. Processing Element)
Data Flow Taxonomy:
• Weight Stationary, Output Stationary
• No local reuse (M-Bottleneck: Roofline)
• Row stationary
Data type: Int8/16 for inference(TPU1), Mix FP16/32 for training(TPU2)
Data Reuse: Conv, Feature map, Filter
Kernel Computation(Toeplitz Matrix, TensorCore, Matrix-Multiply Unit)
Sparse, Compressed Model(EIE, NVIDIA)</td></tr>
</table>

Strategies for DL Acceleration

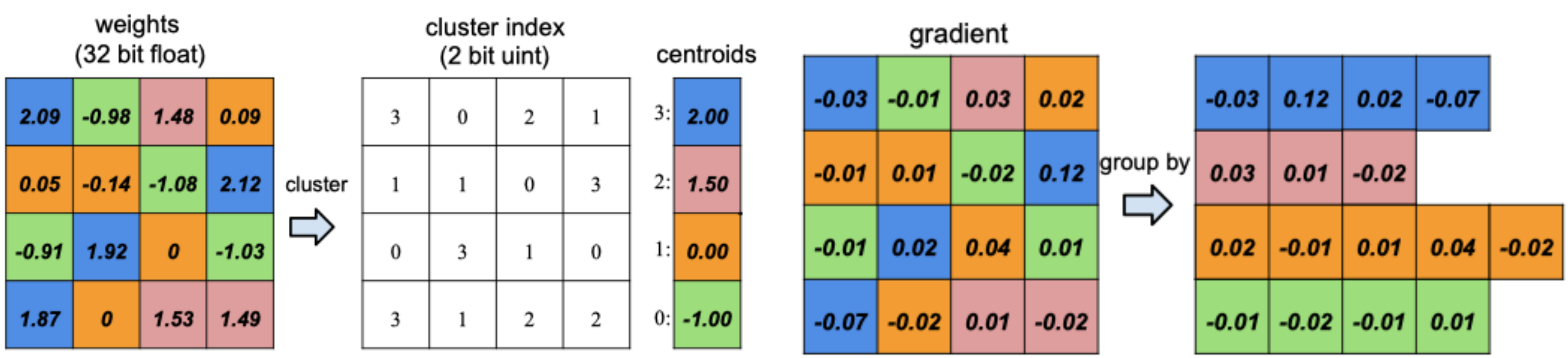
<table>
<tr><th></th><th>Inference</th><th>Training</th></tr>
<tr><td>Algorithm</td><td>Pruning : P-Threshold, +Training
Weight Sharing:
• w-KNN→Quant, Huffman coding
Quantization: Floating P.→ Fixed P.
Low Rank Approximation
Binary/Ternary Net
Computational Transform
• Winograd, FFT, Strassen</td><td>Parallelization:
• Data/Parameter/Hyp.Par
Mixed Precision MAC: FP16 + FP32
Model Distillation:
• Teachers(GGN,VGG,Res)&Student
DSD(Dense-Sparse-Dense) Training
Mixed Precision Net:
• BinaryConnect, QNN</td></tr>
<tr><td>Hardware</td><td colspan="2">Architecture:
• Temporal(SIMD,SIMT with Global Buffer) : General purpose GPUs
• Spatial(Local Memory in Dist. Processing Element)
Data Flow Taxonomy:
• Weight Stationary, Output Stationary
• No local reuse (M-Bottleneck: Roofline)
• Row stationary
Data type: Int8/16 for inference(TPU1), Mix FP16/32 for training(TPU2)
Data Reuse: Conv, Feature map, Filter
Kernel Computation(Toeplitz Matrix, TensorCore, Matrix-Multiply Unit)
Sparse, Compressed Model(EIE, NVIDIA)</td></tr>
</table>

Inference Acceleration Algorithms

Pruning



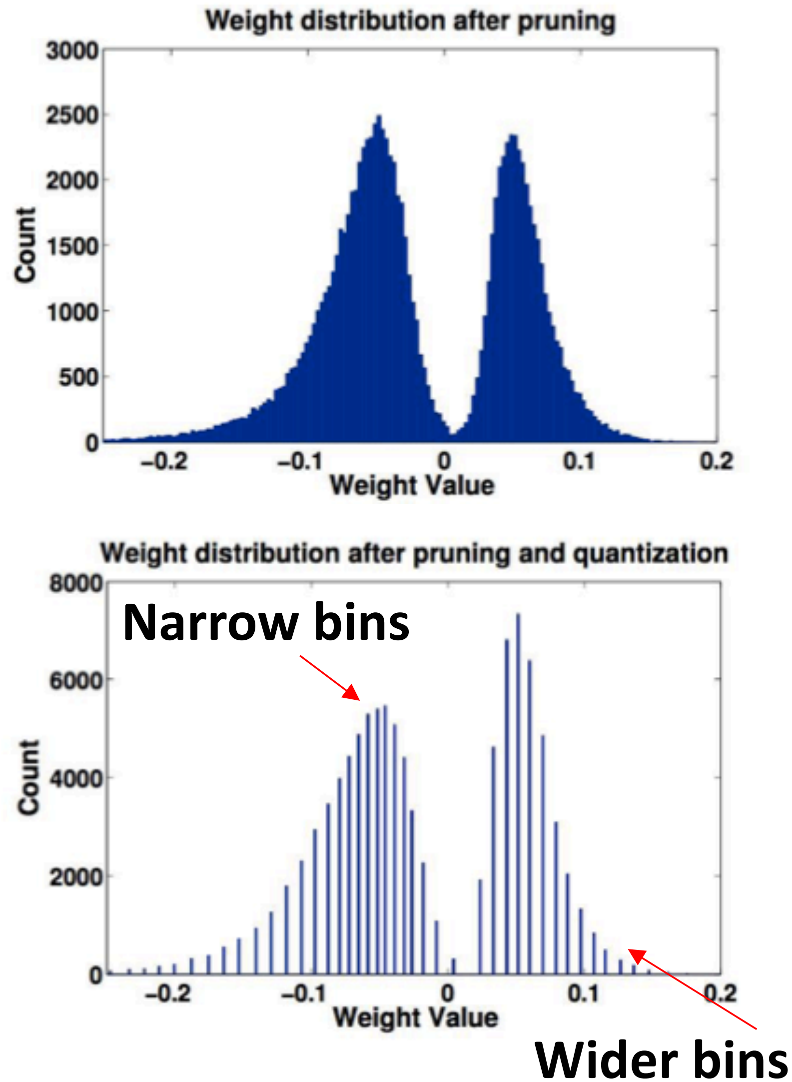
Weight Sharing



Han et al. Deep Compression, ICLR 2016

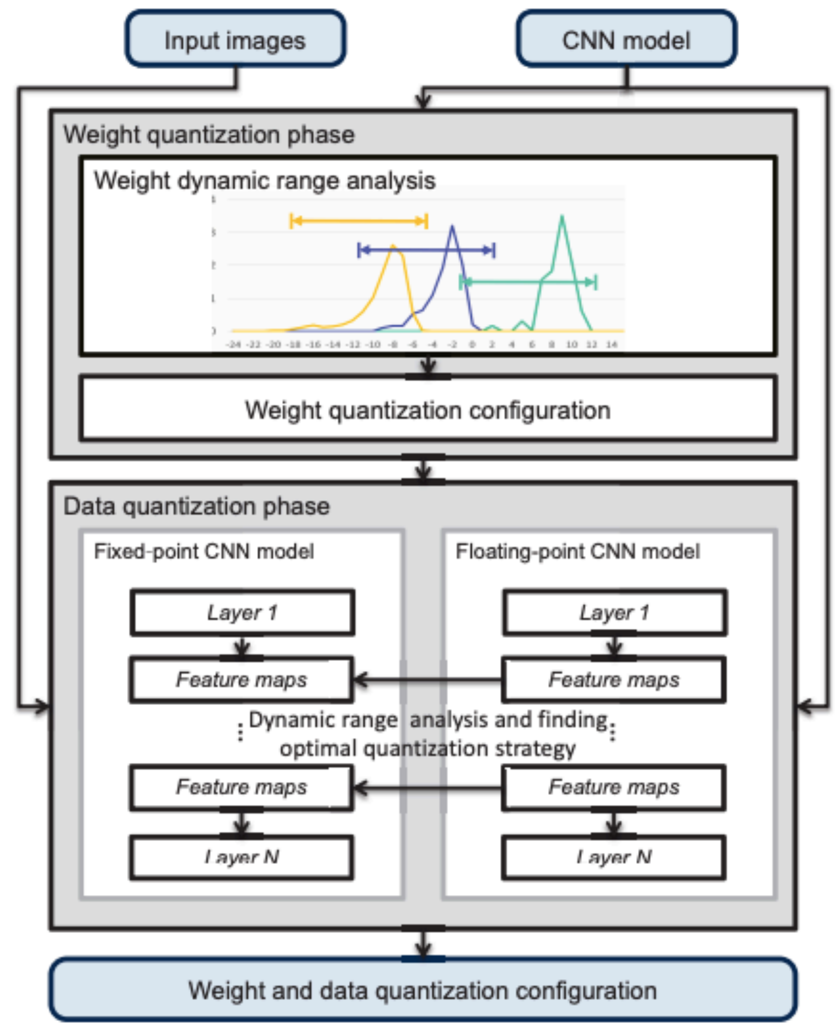
Inference Acceleration Algorithms

Weight Sharing & Quantization



Han et al. Deep Compression, ICLR 2016

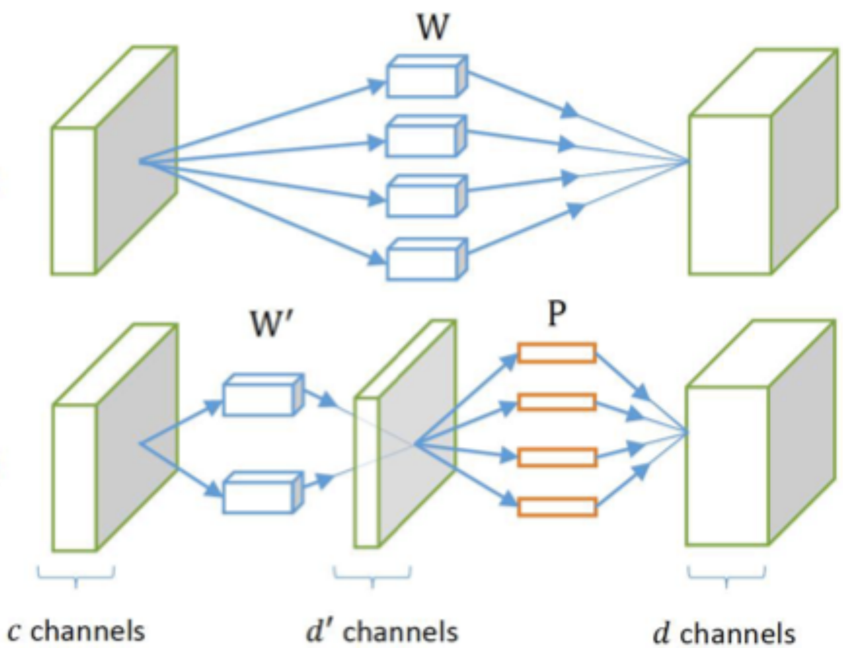
Quantization (Floating→Fixed)



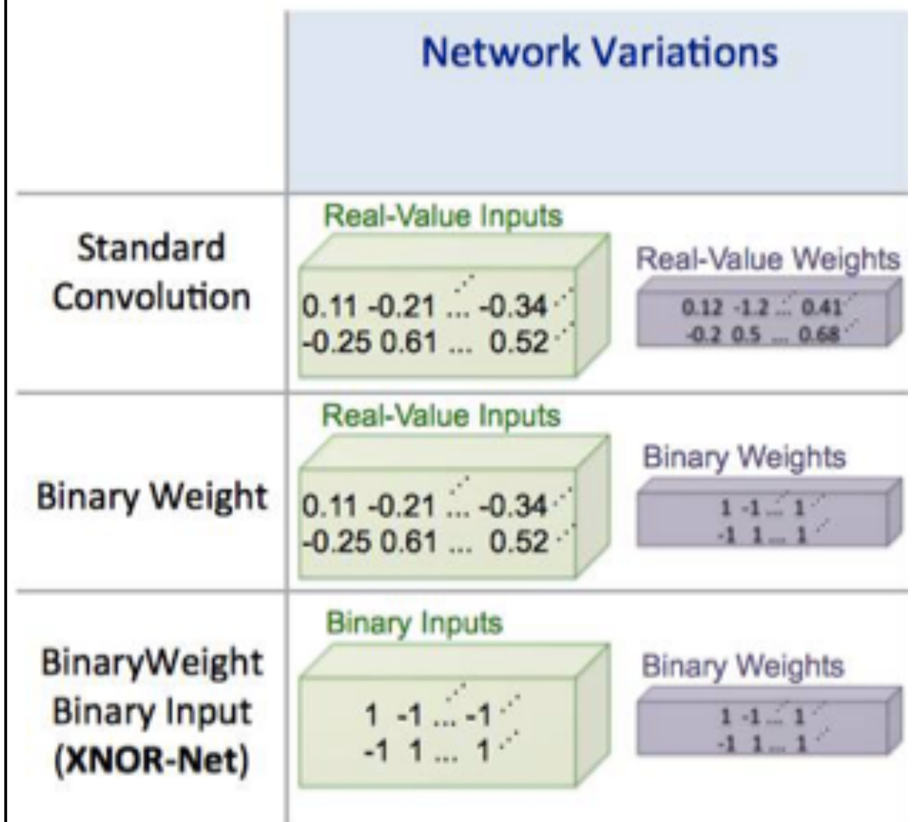
Qiu et al. Going Deeper with Embedded FPGA Platform for Convolutional Neural Network, FPGA'16

Inference Acceleration Algorithms

Low rank approximation



Binary/Ternary Net for Inference(XOR-Net)



Zhang et al Efficient and Accurate Approximations of Nonlinear Convolutional Networks CVPR'15

Rastegari et al. XNOR-Net: ImageNet Classification Using Binary Convolutional Neural Networks ECCV 2016

Inference Acceleration Algorithms

Winograd Transform

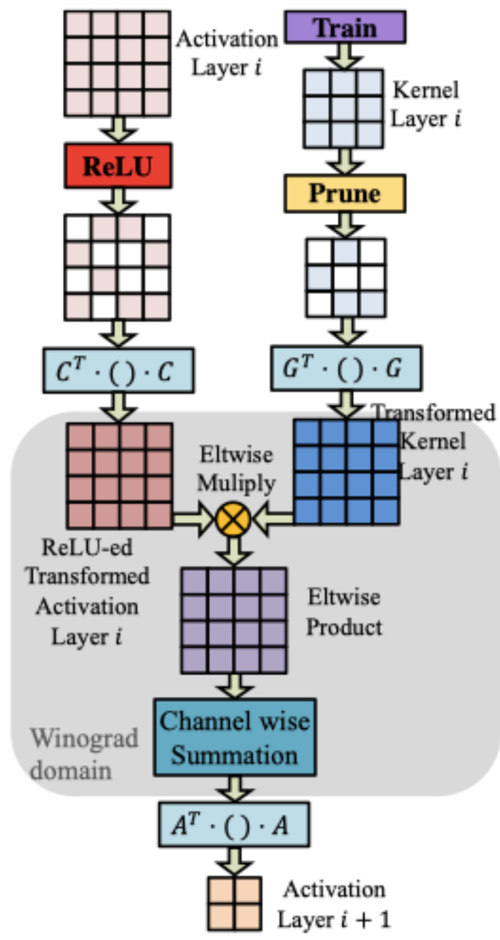
Filter * Input Fmap = Output Fmap

$$\begin{bmatrix} g_{00} & g_{01} & g_{02} \\ g_{10} & g_{11} & g_{12} \\ g_{20} & g_{21} & g_{22} \end{bmatrix} * \begin{bmatrix} d_{00} & d_{01} & d_{02} & d_{03} \\ d_{10} & d_{11} & d_{12} & d_{13} \\ d_{20} & d_{21} & d_{22} & d_{23} \\ d_{30} & d_{31} & d_{32} & d_{33} \end{bmatrix} = \begin{bmatrix} y_{00} & y_{01} \\ y_{10} & y_{11} \end{bmatrix}$$

3x3 filter & 4x4 inputs
= 9x4 = **36 multiplications**

Winograd Trasform
→ 4x4 filter & 4x4 inputs
= **16 element-wise multiplications**

Winograd Transform & Prune



(a)

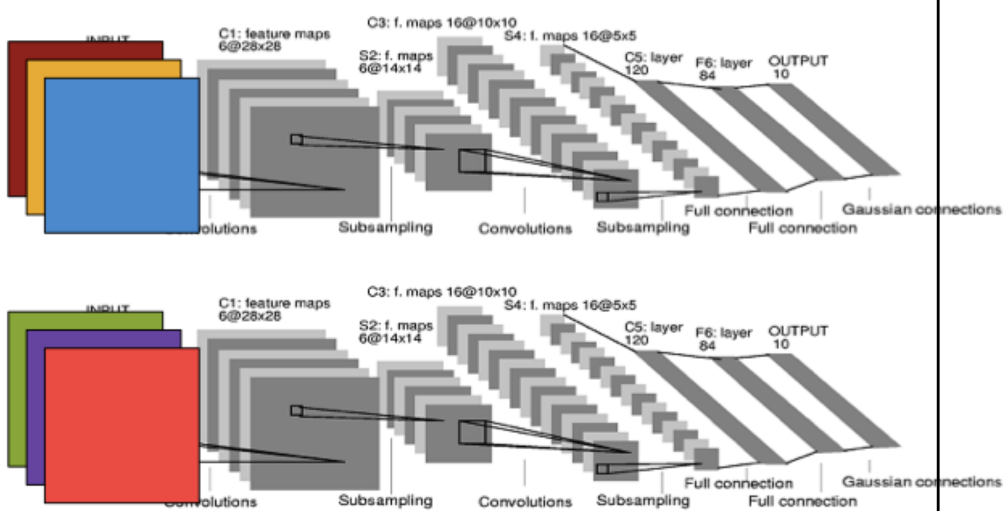
A. Lavin et. al., "Fast Algorithms for Convolutional Neural Networks", arXiv 2015
Liu et al. Figure "Efficien Sparse-Winograd Winograd Convolutional convolution with Neural sparse Networks", ICLR 2017

Strategies for DL Acceleration

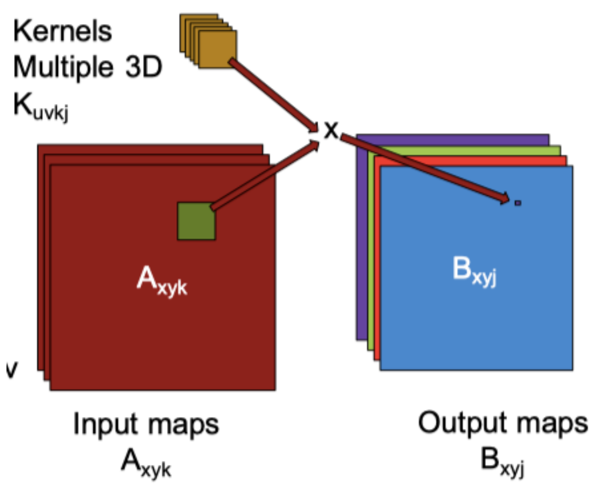
	Inference	Training
Algorithm	**Pruning** : P-Threshold, +Training **Weight Sharing**: • w-KNN→Quant, Huffman coding **Quantization**: Floating P.→ Fixed P. **Low Rank Approximation** **Binary/Ternary Net** **Computational Transform** • Winograd, FFT, Strassen	**Parallelization**: • Data/Parameter/Hyp.Par **Mixed Precision MAC:** FP16 + FP32 **Model Distillation**: • Teachers(GGN,VGG,Res)&Student DSD(Dense-Sparse-Dense) Training **Mixed Precision Net:** • BinaryConnect, QNN
Hardware	**Architecture:** • Temporal(SIMD,SIMT with Global Buffer) : General purpose GPUs • Spatial(Local Memory in Dist. Processing Element) **Data Flow Taxonomy:** • Weight Stationary, Output Stationary • No local reuse (M-Bottleneck: Roofline) • Row stationary **Data type**: Int8/16 for inference(TPU1), Mix FP16/32 for training(TPU2) **Data Reuse**: Conv, Feature map, Filter **Kernel Computation**(Toeplitz Matrix, TensorCore, Matrix-Multiply Unit) **Sparse, Compressed Model**(EIE, NVIDIA)	

Training Acceleration Algorithms

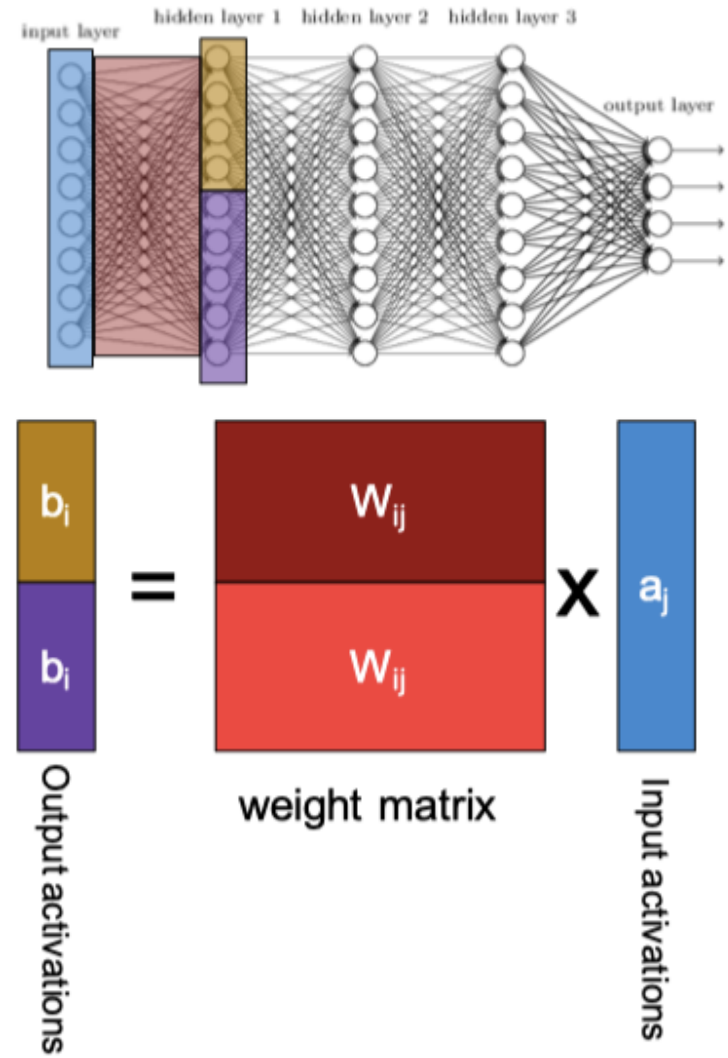
Data Parallelization



Model Parallelization

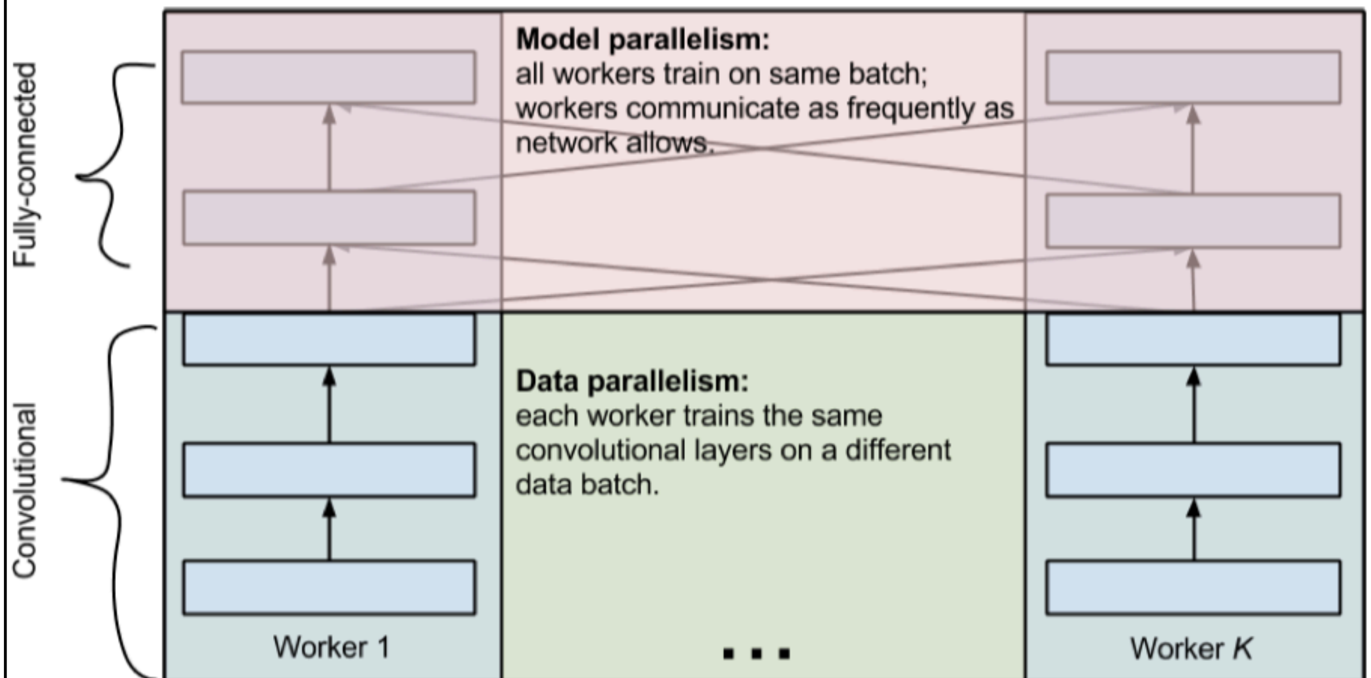


FC Model Parallelization



Training Acceleration Algorithms

Alex's Weird CNN Parallelization Trick



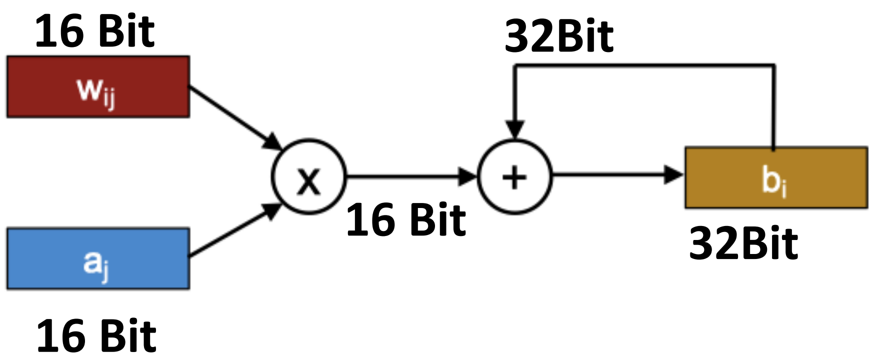
Metrics	LeNet-5	AlexNet
Top-5 error	n/a	16.4
Input Size	28x28	227x227
# of CONV Layers	**2**	**5**
Filter Sizes	5	3, 5,11
# of Channels	1, 6	3 - 256
# of Filters	6, 16	96 - 384
Stride	1	1, 4
# of Weights	2.6k	2.3M
# of MACs	283k	666M
# of FC layers	**2**	**3**
# of Weights	58k	58.6M
# of MACs	58k	58.6M
Total Weights	**60k**	**61M**
Total MACs	**341k**	**724M**

- **Most of Parameters → FC**
- **Most of MACs→ Filter**

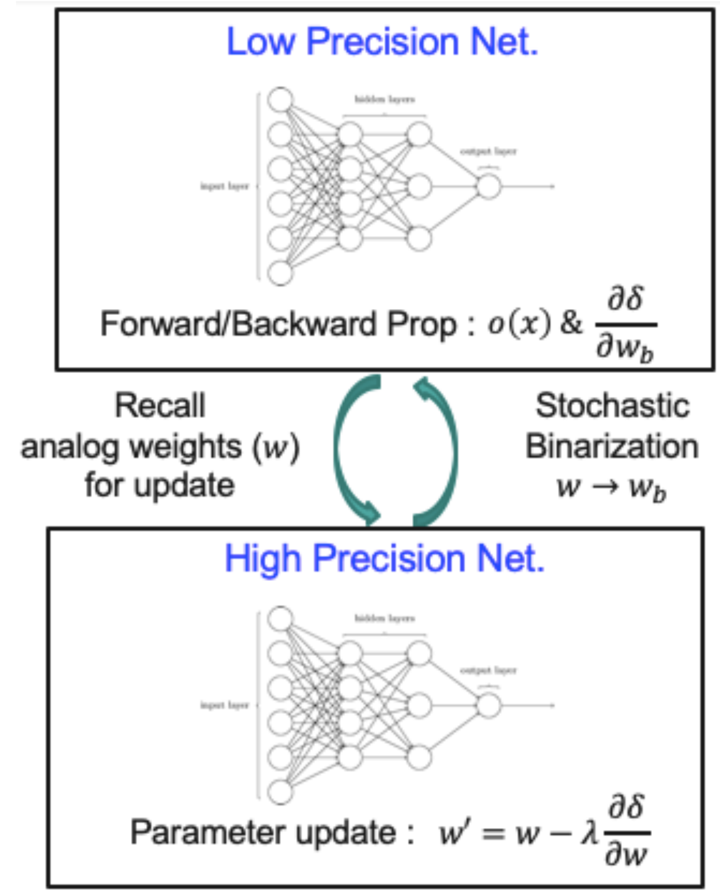
A. Krizhevsky, "One weird trick for parallelizing convolutional neural networks",2014

Training Acceleration Algorithms

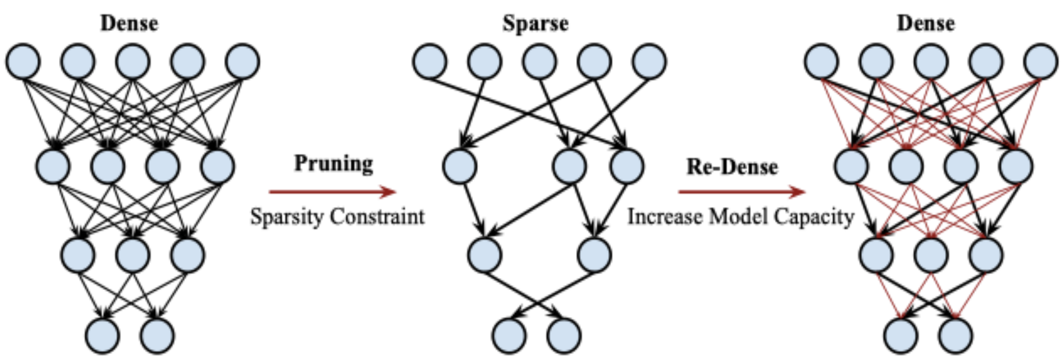
Mixed Precision MAC



Mixed Precision MAC



Dense-Sparse-Dense Training



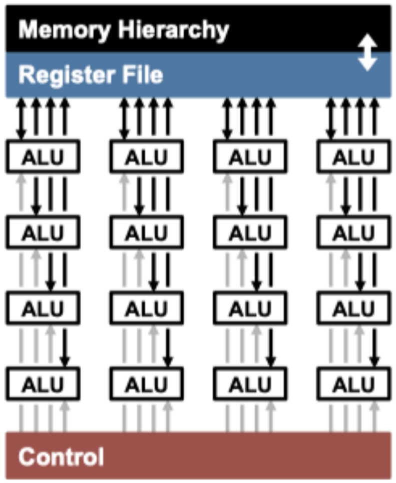
W. Dally, NIPS2015 & M. Courbariaux et. al., 2016

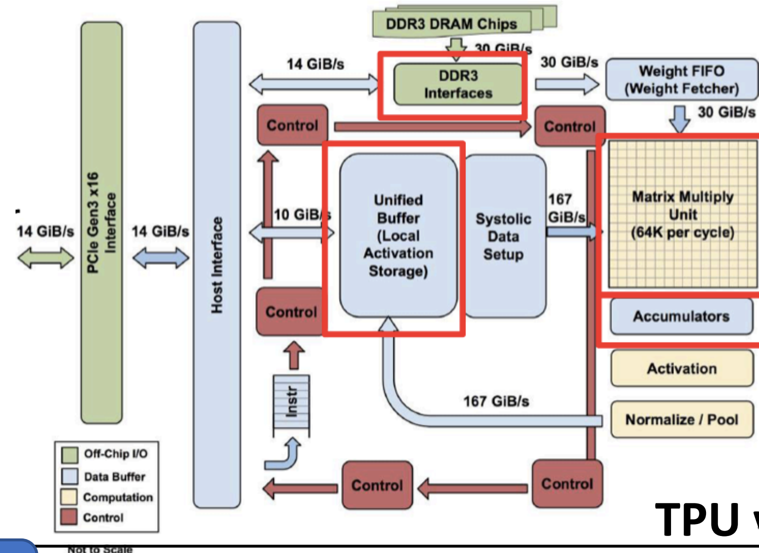
Strategies for DL Acceleration

	Inference	Training
Algorithm	**Pruning** : P-Threshold, +Training **Weight Sharing**: • w-KNN→Quant, Huffman coding **Quantization**: Floating P.→ Fixed P. **Low Rank Approximation** **Binary/Ternary Net** **Computational Transform** • Winograd, FFT, Strassen	**Parallelization**: • Data/Parameter/Hyp.Par **Mixed Precision MAC:** FP16 + FP32 **Model Distillation**: • Teachers(GGN,VGG,Res)&Student DSD(Dense-Sparse-Dense) Training **Mixed Precision Net:** • BinaryConnect, QNN
Hardware	**Architecture:** • Temporal(SIMD,SIMT with Global Buffer) : General purpose GPUs • Spatial(Local Memory in Dist. Processing Element) **Data Flow Taxonomy:** • Weight Stationary, Output Stationary • No local reuse (M-Bottleneck: Roofline) • Row stationary **Data type**: Int8/16 for inference(TPU1), Mix FP16/32 for training(TPU2) **Data Reuse**: Conv, Feature map, Filter **Kernel Computation**(Toeplitz Matrix, TensorCore, Matrix-Multiply Unit) **Sparse, Compressed Model**(EIE, NVIDIA)	

Hardware DL Acceleration

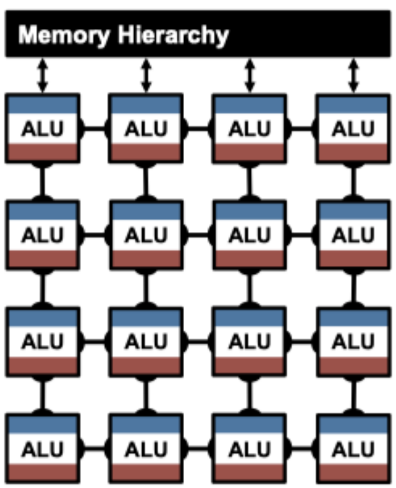
Temporal Architectures (SIMD, SIMT)

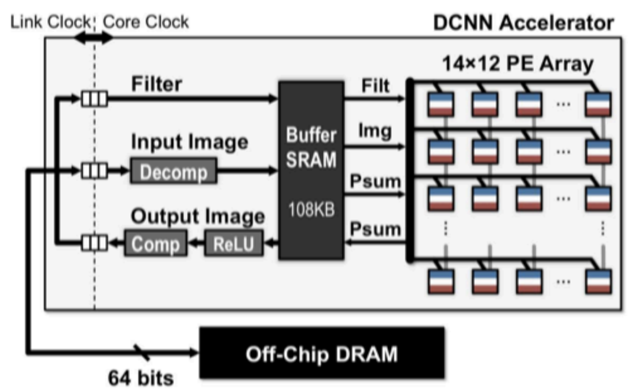




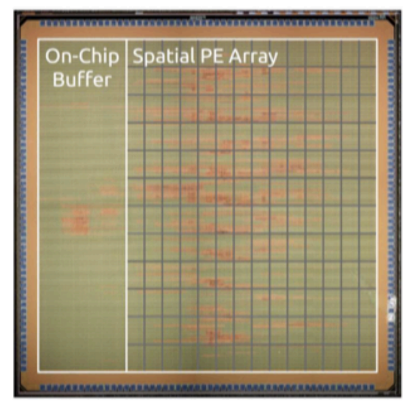
TPU v.1

Spatial Architectures (Local Memory & PE)





Eyeriss Architecture



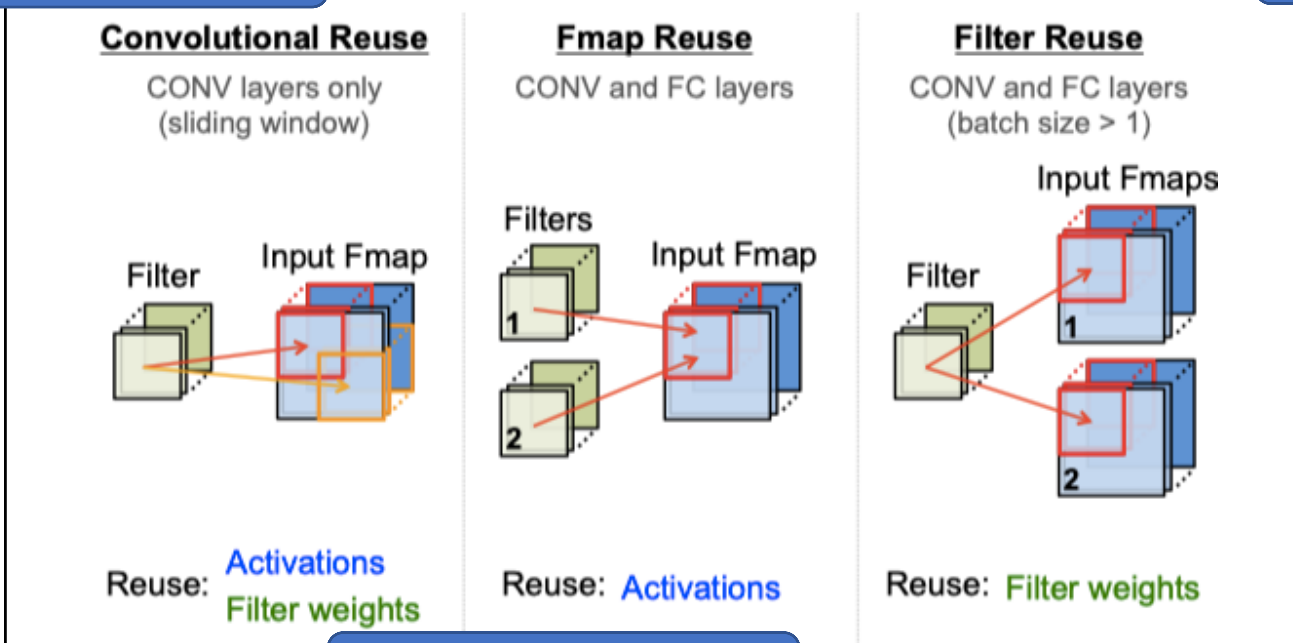
Die Photo

MIT:Eyeriss

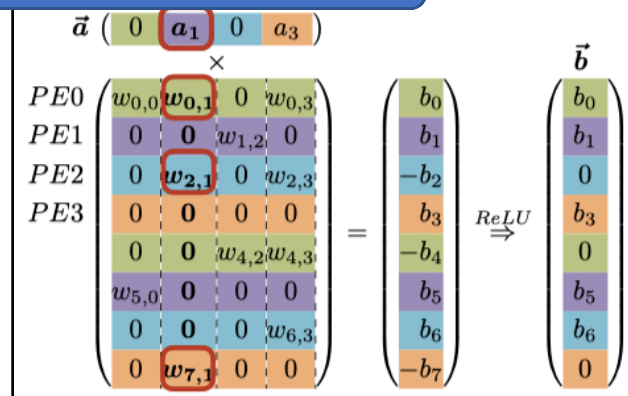
TPU, Google & V. Sze et. al., 2017

Hardware DL Acceleration

Data Reuse

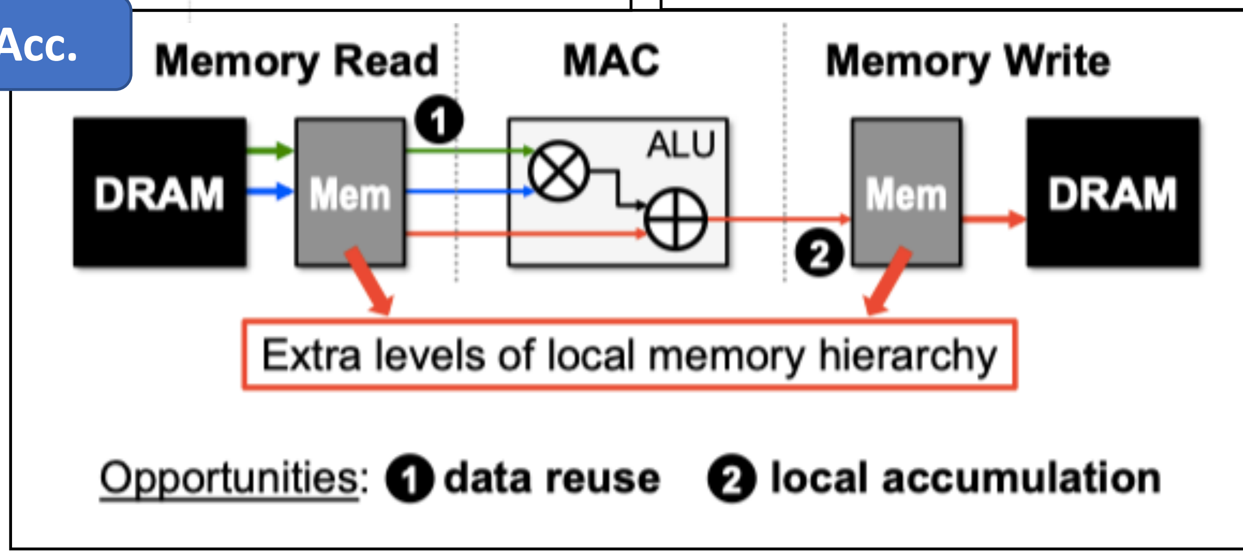


Exploiting Sparsity



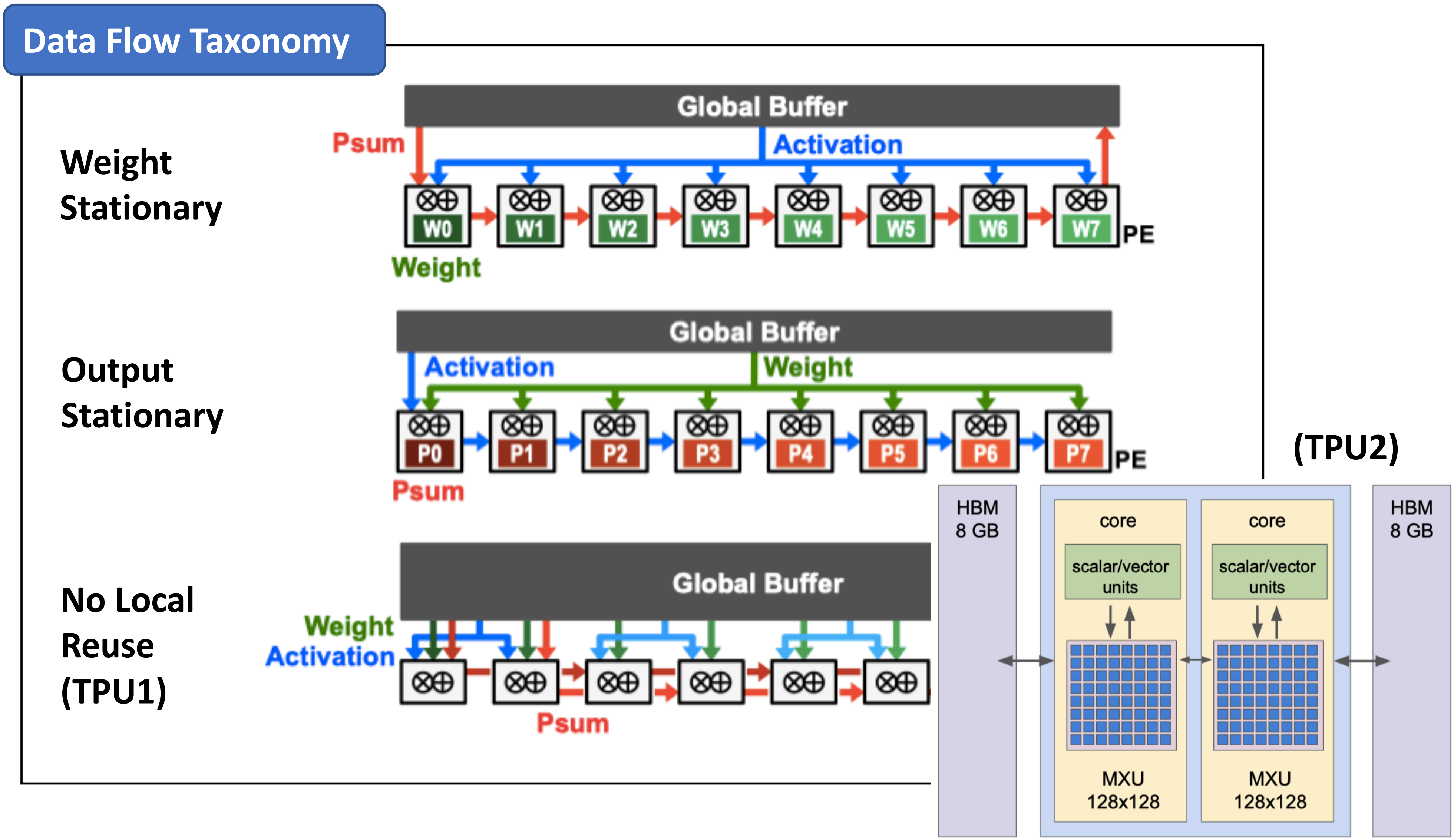
Virtual Weight	$W_{0,0}$	$W_{0,1}$	$W_{4,2}$	$W_{0,3}$	$W_{4,3}$
Relative Index	0	1	2	0	0
Column Pointer	0	1	2	3	

Reuse & Local Acc.



V. Sze et. al., 2017, S. Han et. al., 2017.

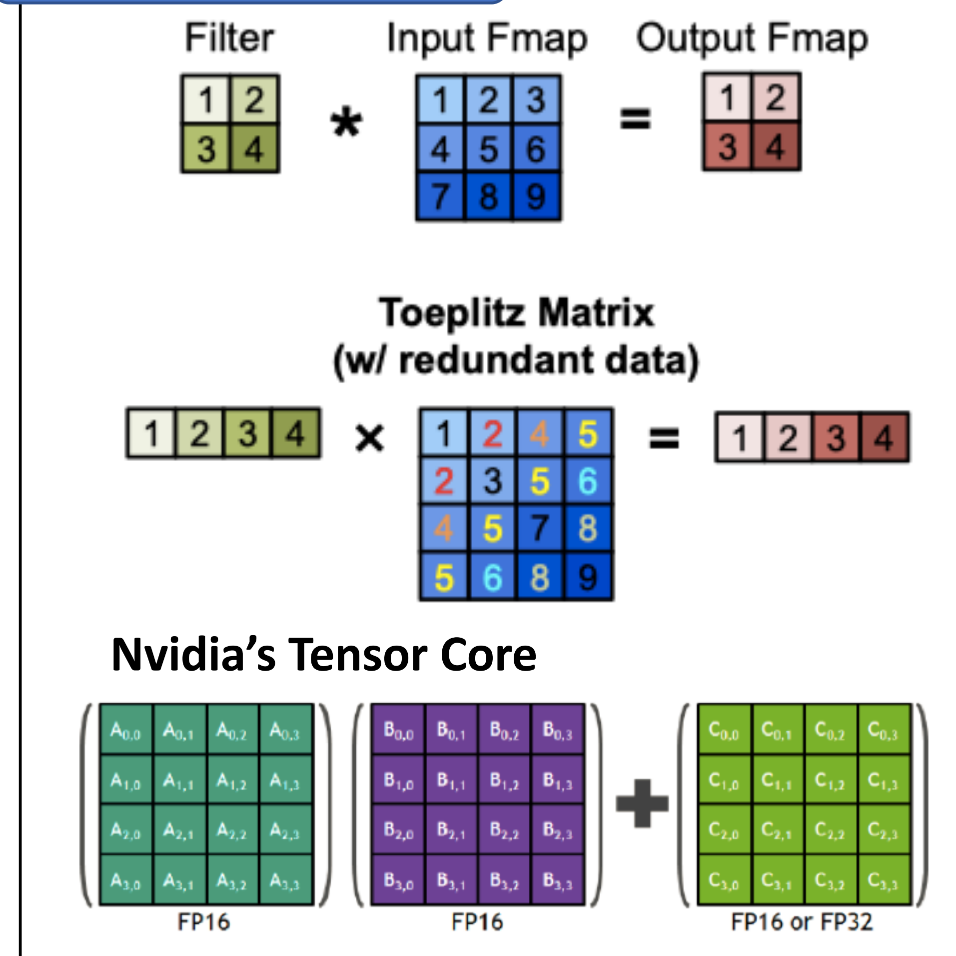
Hardware DL Acceleration



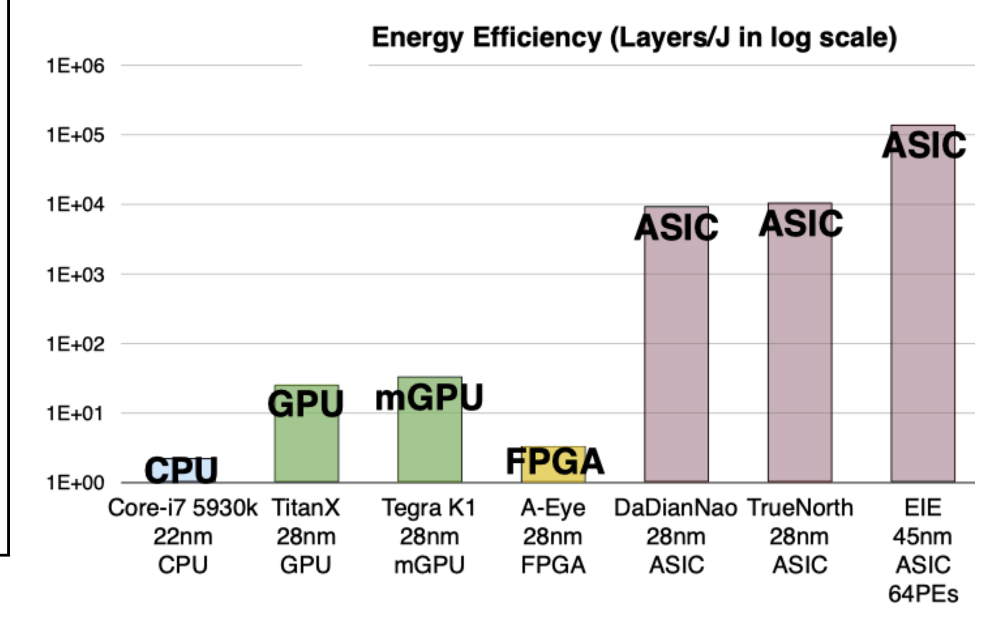
V. Sze et. al., 2017 & J.Dean, NIPS 2017.

Hardware DL Acceleration

Kernal Computation



Energy Efficiency



V. Sze et. al., 2017 & S. Han, 2017.

Strategies for DL Acceleration

	Inference	Training
Algorithm	**Pruning** : P-Threshold, +Training **Weight Sharing**: • w-KNN→Quant, Huffman coding **Quantization**: Floating P.→ Fixed P. **Low Rank Approximation** **Binary/Ternary Net** **Computational Transform** • Winograd, FFT, Strassen	**Parallelization**: • Data/Parameter/Hyp.Par **Mixed Precision MAC:** FP16 + FP32 **Model Distillation**: • Teachers(GGN,VGG,Res)&Student DSD(Dense-Sparse-Dense) Training **Mixed Precision Net:** • BinaryConnect, QNN
Hardware	**Architecture:** • Temporal(SIMD,SIMT with Global Buffer) : General purpose GPUs • Spatial(Local Memory in Dist. Processing Element) **Data Flow Taxonomy:** • Weight Stationary, Output Stationary • No local reuse (M-Bottleneck: Roofline) • Row stationary **Data type**: Int8/16 for inference(TPU1), Mix FP16/32 for training(TPU2) **Data Reuse**: Conv, Feature map, Filter **Kernel Computation**(Toeplitz Matrix, TensorCore, Matrix-Multiply Unit) **Sparse, Compressed Model**(EIE, NVIDIA)	